Application Development Domain

Technical Architecture

Appendix C

Microsoft .NET Migration Guidelines

The following recommendations for State adoption of EWTA-approved Microsoft .NET technologies are listed in descending order based on estimated overall best value (benefit : cost ratio):

Recommended Strategies:

1. Adapt infrastructure to begin hosting applications based on the .NET framework.

Rationale: *Since .NET applications can coexist with existing Microsoft-based applications on the same Windows 2000 servers by installing a free download (.NET Framework), hosting .NET applications has the potential to offer a large value with a relatively small investment.*

Approach: *Allow vendors to bid software solutions based on EWTA-compliant Microsoft .NET technologies to be run on the State's existing infrastructure.*

2. Use ASP.NET to develop new applications and expand[1] existing ASP applications.

Rationale: *Developing Custom In-House Web applications using ASP.NET offers the greatest potential to both increase developer productivity and application performance without incurring significant deployment costs.*

Approach: *Give priority to training those developers currently doing ASP development to help them migrate to ASP.NET realizing significant gains in productivity.*

3. **Use VB.NET to develop new applications and expand existing Visual Basic 6.0 applications.**

Rationale: *Developing Custom In-House applications using VB.NET offers significant increases in developer productivity after the initial learning curve. Strong interoperability makes it possible to re-use existing VB (COM) components.*

Approach: *Visual Basic V 6.0 developers can transition gradually to .NET by first getting used to Visual Studio.NET (Visual Studio V6.0 is still required to maintain your Visual Basic 6.0 applications). The new VB.NET programmers can then (with proper training) gradually begin introducing .NET components or new modules into their applications. This incremental approach minimizes risk and allows developers to learn the .NET technologies at their own pace.*

Alternately experienced Visual Basic V 6.0 developers can begin new applications using VB.NET (recommend starting with relatively simple, non-time critical applications to minimize risk).

[1] While ASP.NET and ASP applications can co-exist on the same servers, they will not automatically share state (session variables). To share state between these two technologies, persist state using a database (e.g. SQL Server).

4. Re-write existing (Visual Basic 6.0) systems using .NET technologies.

Rationale: *It may make sense to convert selected VB 6.0 applications to VB.NET using the freely available Visual Basic 6.0 to VB.NET migration tool (95% effective)*

Approach: *Do not bother for systems with only a few years of useful life.*
For long-lived systems it may make sense to progressively introduce .NET components and port code/logic when existing (COM+) components need significant rework or modification because of other development changes. Use this approach for mission important/critical systems whose expected lifetime is beyond February 2008 when Visual Basic 6.0 support ceases, to take advantage of .NET features when changes are required, and to avoid the possibility of a future shortage of legacy Visual Basic 6.0 developers. The more dynamic the business model the application supports, the more the application would benefit from migration to .NET. Alternately, consider buying a packaged application or leveraging an existing State application – migration is not trivial.

5. Re-write existing ASP applications using ASP.NET.

Rationale: *Convert ASP applications to .NET only when there is a compelling reason to do so.*

Approach: *The radical difference in approach between ASP (script mixed with HTML) and ASP.NET (compiled code behind HTML pages) and the lack of a migration tool make this a difficult conversion. Try coexistence, Strategy 2 above, first unless the application is simple.*